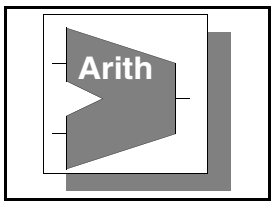
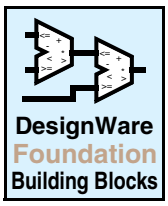




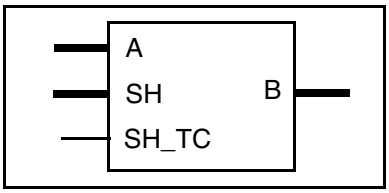
DW_rbsh

Barrel Shifter with Preferred Right Direction

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

- Parameterized data and shift coefficient word lengths
- Capable of rotating in both directions
- Inferable using a function call



Description

DW_rbsh is a general-purpose barrel shifter that prefers to rotate the information in the right direction. The input data `A` is rotated right or left by the number of bit positions specified by the control input `SH`. Rotation implies that bits shifted out wrap around from the LSB to the MSB when it is a right shift, and from the MSB to the LSB when it is a left shift.The recommended maximum value of the parameter *SH_width* is related to *A_width* by the equation:

```
SH_width = ceil(log2[A_width])
```

Smaller values of *SH_width* can be used to save hardware. *SH_width* can be larger than `ceil(log2[A_width])`, but this choice usually generates unnecessary hardware.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*A_width*	Input	Input data
SH	*SH_width*	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control ■ 0 = Unsigned ■ 1 = Signed
B	*A_width*	Output	Shifted data out

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	≥ 1 (see Table 1-4 on page 2)	Word length of `A` and `B`
SH_width	≤ ceil($\log_2$[*A_width*])	Word length of `SH`

Table 1-3 Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
str	Synthesis model target for speed	DesignWare
astr	Synthesis model target for area	DesignWare
mx2	Implement using 2:1 multiplexers only The mx2 implementation is valid only for *SH_width* values up to and including 31.	none

a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

The following table lists some values for A_width and the corresponding *SH_width* values. For example, if *A_width* = 8, *SH_width* = 3 is adequate because there can be at most $2^3 = 8$ shift combinations.

When `SH_TC` = 0, the coefficient SH is interpreted as an unsigned positive number and DW_rbsh performs only right rotate operations.

When `SH_TC` = 1, `SH` is interpreted as a two's complement number. A negative `SH` value performs a left rotate and a positive `SH` value performs a right rotate.

Table 1-4 Sample Parameter Values

A_width	SH_width
1 - 2	1
3 - 4	2
5 - 8	3
9 - 16	4
17 - 32	5
33 - 64	6

Table 1-5 Simulation Models

Model	Function
dw/dw01/src/DW_rbsh_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_rbsh.v	Verilog simulation model source code

Table 1-6 Truth Table (*A_width* = 8, *SH_width* = 3)

SH(2:0)	SH_TC	B(7)	B(6)	B(5)	B(4)	B(3)	B(2)	B(1)	B(0)
000	X	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)
001	X	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)
010	X	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
011	X	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)
100	0	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)
101	0	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)
110	0	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)
111	0	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)
100	1	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)
101	1	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)
110	1	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)
111	1	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)

Related Topics

- Logic – Combinational Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Function Inferencing - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_arith.all;

entity DW_rbsh_func is
      generic (
        func_A_width : POSITIVE := 8;
        func_SH_width : POSITIVE := 3
        );
      port (
        func_A : in std_logic_vector(func_A_width-1 downto 0);
        func_SH : in std_logic_vector(func_SH_width-1 downto 0);
        func_SH_TC : in std_logic;
        B_func : out std_logic_vector(func_A_width-1 downto 0)
        );
    end DW_rbsh_func;


architecture func of DW_rbsh_func is

begin

    -- Functional inference of DW_rbsh
    B_func <= std_logic_vector(DWF_rbsh(UNSIGNED(func_A),UNSIGNED(func_SH)));

end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW_rbsh_func( func_A, func_SH, func_SH_TC, B_func );

parameter func_A_width = 8;
parameter func_SH_width = 3;

// secondary parameters used to pass parameters to function
// when parameter names differ

parameter A_width = func_A_width;
parameter SH_width = func_SH_width;

// Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
// to your .synopsys_dc.setup file (for synthesis) and add
// +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
// (for simulation).
`include "DW_rbsh_function.inc"

input [func_A_width-1 : 0] func_A;
input [func_SH_width-1 : 0] func_SH;
input func_SH_TC;
output [func_A_width-1 : 0] B_func;

    // Infer DW_rbsh

     assign B_func = DWF_rbsh_uns(func_A,func_SH);


endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.dw_foundation_comp.all;

entity DW_rbsh_inst is
      generic (
        inst_A_width : POSITIVE := 8;
        inst_SH_width : POSITIVE := 3
        );
      port (
        inst_A : in std_logic_vector(inst_A_width-1 downto 0);
        inst_SH : in std_logic_vector(inst_SH_width-1 downto 0);
        inst_SH_TC : in std_logic;
        B_inst : out std_logic_vector(inst_A_width-1 downto 0)
        );
    end DW_rbsh_inst;

architecture inst of DW_rbsh_inst is

begin

    -- Instance of DW_rbsh
    U1 : DW_rbsh
    generic map (
          A_width => inst_A_width,
          SH_width => inst_SH_width
          )
    port map (
          A => inst_A,
          SH => inst_SH,
          SH_TC => inst_SH_TC,
          B => B_inst
          );

end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_rbsh_inst( inst_A, inst_SH, inst_SH_TC, B_inst );

parameter A_width = 8;
parameter SH_width = 3;


input [A_width-1 : 0] inst_A;
input [SH_width-1 : 0] inst_SH;
input inst_SH_TC;
output [A_width-1 : 0] B_inst;

    // Instance of DW_rbsh
    DW_rbsh #(A_width, SH_width) U1 (
                .A(inst_A),
                .SH(inst_SH),
                .SH_TC(inst_SH_TC),
                .B(B_inst) );

endmodule
```

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com